Proceedings – 23rd Annual Conference – IEEE/EMBS Oct.25-28, 2001, Istanbul, TURKEY

LEARN++: AN INCREMENTAL LEARNING ALGORITHM BASED ON PSYCHO-PHYSIOLOGICAL MODELS OF LEARNING

R. Polikar
Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028
polikar@rowan.edu

Abstract- **An incremental learning algorithm, Learn++, which allows supervised classification algorithms to learn from new data without forgetting previously acquired knowledge, is introduced. Learn++ is based on generating multiple classifiers using strategically chosen distributions of the training data and combining these classifiers through weighted majority voting. Learn++ shares various notions with psycho-physiological models of learning. The Learn++ algorithm, simulation results, and how the algorithm is related to various concepts in psycho-physiological learning models are discussed.**

Keywords **- Incremental learning, ensemble of classifiers, short and long term memory, psycho-physiological models of learning.**

I. INTRODUCTION

Various paradigms have been developed over the last few decades to emulate learning and decision-making capabilities of the brain. In particular, artificial neural networks (ANNs) have been developed based on neurophysiological models, with one-to-one association between learning and training, neurons and nodes, synaptic connections and weights, etc.

One of the shortcomings of many popular neural network architectures, however, is their inability to learn additional information incrementally from new data that become available after the original training session. Multilayer perceptron (MLP), radial basis function neural networks, wavelet networks, probabilistic neural networks, Kohonen networks and most self-organizing maps are all examples of popular ANN paradigms that are not capable of incremental learning. When additional data become available, these networks are typically discarded and retrained with combined data, which consists of both the original and the new data. All previously acquired knowledge is therefore lost, a phenomenon commonly known as *catastrophic forgetting*. Furthermore, if the original data is no longer available at the time new data become available, or when new data includes instances from new classes not present in the original data, these algorithms become completely useless for learning new information.

The brain, however, does not suffer from catastrophic forgetting, and is capable of learning new information without forgetting previously acquired knowledge, even in the absence of the source of the original information. For example, a student learning new vocabulary does not necessarily forget previously learned vocabulary, nor does s/he need to re-study all vocabulary previously mastered to learn new vocabulary.

Learn++, first introduced in [1], is an algorithm that allows any supervised classification algorithm to learn incrementally from new data in the absence of original (or all previous) data. Furthermore, Learn++ can also learn additional classes that may be introduced with the new data. This paper describes this algorithm and points out the similarities between the algorithm and various concepts in psycho-physiological models of learning.

II. LEARN++ FOR INCREMENTAL LEARNING

Learn++ is based on generating an ensemble of classifiers using different distributions of a training dataset. The training dataset is updated as new training databases become available. The final classification is then made by a weighted majority voting of classifier outputs. Combining classifiers using such a voting scheme has been used effectively in improving classifier performance. In particular, Schapire [2] has demonstrated that a strong classification algorithm can be generated from an ensemble of weak classification algorithms through a procedure called *boosting*. Using boosting, he showed that when weak classifiers are combined using a weighted majority voting, their combined performance is better than the best classifier in the ensemble, and comparable to (or better than) the performance of a strong classifier. Furthermore, this approach typically results in faster training than conventional strong classifiers, and more importantly, it does not suffer from overtraining. Learn++ is inspired by AdaBoost [3], which itself is based on boosting and was originally developed for improving classification performance of a weak classifier. Learn++, however, differs from AdaBoost in the sense that it is designed to add incremental learning capability to an existing classifier. Learn++ is described in Fig. 1.

Inputs to Learn++ are the training data S_k of m samples randomly selected from the currently available database $\mathcal{D}_k$, a weak learning algorithm **WeakLearn**, and an integer T_k, specifying the number of classifiers to be generated. Neural networks are especially suited for this setup, since they qualify as weak learners, where their weakness can be controlled by the number of layers/nodes and/or error goal. Learn++ requires that each weak learner be able to correctly classify at least 50% of the current training set, generating only a rough estimate of the actual decision boundary. Therefore, each weak learner can be trained very quickly, whereas strong learners typically spend a majority of their training time in fine-tuning the decision boundary.

Each classifier can be thought of as a hypothesis h from the input space X to the output space Y. Learn++ asks **WeakLearn** to generate multiple hypotheses using different subsets of the training data S_k, and each hypothesis learns only a portion of the input space. This is achieved by iteratively updating a distribution D_t, $t=1,2,\ldots, T_k$ from which training subsets are chosen. The distribution itself is obtained by normalizing a set of weights assigned to each instance based on the classification performance of the classifiers on that instance (step 1). In general, instances that are difficult to classify are given higher weights to increase their chance of being selected into the next training dataset. The weights for the first iteration $w_1(i)$ (and, hence, the first distribution D_1) are initialized to $1/m$, $i=1,2,\ldots,m$ giving equal likelihood for each instance to be selected into the first training subset.

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
25OCT2001	N/A	-

Title and Subtitle Learn: An Incremental Learning Algorithm Based on Psycho-Physiological Models of Learning	**Contract Number**
	Grant Number
	Program Element Number
Author(s)	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es) US Army Research, Development & Standardization Group (UK) PSC 802 Box 15 FPO AE 09499-1500	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)

Distribution/Availability Statement
Approved for public release, distribution unlimited

Supplementary Notes
Papers from the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, October 25-28, 2001, held in Istanbul, Turkey. See also ADM001351 for entire conference on cd-rom., The original document contains color images.

Abstract

Subject Terms

Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU

Number of Pages
4

Input: For each dataset drawn from $\mathcal{D}_k$ $k=1,2,\ldots,K$

- Sequence of m examples $S=[(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)]$.
- Weak learning algorithm **WeakLearn**.
- Integer T_k, specifying the number of iterations.

Do for each $k=1,2,\ldots,K$:

Initialize $w_1(i) = D_1(i) = 1/m, \ \forall i, \ i = 1,2,\cdots,m$

Do for $t = 1,2,\ldots,T_k$:

1. Set $D_t = \mathbf{w}_t \Big/ \sum_{i=1}^{m} w_t(i)$ so that D_t is a distribution.
2. Randomly choose training TR_t and testing TE_t subsets from D_t.
3. Call **WeakLearn**, providing it with TR_t from distribution D_t.
4. Get back a hypothesis $h_t : X \rightarrow Y$, and calculate the error of h_t : $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ on $TR_t + TE_t$. If $\varepsilon_t > ½$, set $t = t - 1$, discard h_t and go to step 2. Otherwise, compute normalized error as $\beta_t = \varepsilon_t / (1-\varepsilon_t)$.
5. Call weighted majority, obtain the overall hypothesis $H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log(1/\beta_t)$, compute composite error $E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)[\![H_t(x_i) \neq y_i]\!]$. If $E_t > ½$ set $t = t - 1$, discard h_t and go to step 2.
6. Set $B_t = E_t/(1-E_t)$, and update the weights:

$$w_{t+1}(i) = w_t(i) \times \begin{cases} B_t, & if \ \ H_t(x_i) = y_i \\ 1, & otherwise \end{cases} = w_t(i) \times B_t^{\,1-[\![H_t(x_i)\neq y_i]\!]}$$

Call Weighted majority and **Output** the final hypothesis:

$$H_{final}(x) = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:h_t(x)=y} \log\frac{1}{\beta_t}$$

Figure 1. Algorithm LEARN++

At each iteration $t=1,2,\ldots,T_k$, Learn++ first dichotomizes S_k into a training subset TR_t and a test subset TE_t according to D_t (step 2), and calls **WeakLearn** to generate the hypothesis $h_t : X \rightarrow Y$ (step 3). The error of h_t on $S_k = TR_t + TE_t$ is computed as (step 4)

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{1}$$

If $\varepsilon_t > ½$, h_t is discarded and a new TR_t and TE_t are selected. Otherwise, the normalized error β_t is computed as

$$\beta_t = \varepsilon_t / 1 - \varepsilon_t \tag{2}$$

All hypotheses that are generated in the previous t iterations are then combined using weighted majority voting (step 5). The voting weights are based on the normalized errors β: hypotheses with lower normalized errors are given larger weights. A classification decision is then made based on the outputs of individual hypotheses, which constitutes the *composite hypothesis* H_t

$$H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log\frac{1}{\beta_t} \tag{3}$$

Note that H_t decides on the class that is selected by the (weighted) majority of all t hypotheses. The error made by H_t is then computed as

$$E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)[\![H_t(x_i) \neq y_i]\!] \tag{4}$$

where $[\![\cdot]\!]$ is 1 if the predicate holds true, and 0 otherwise. If $E_t > ½$, current h_t is discarded, a new training subset is selected and a new h_t is generated. This, however, is unlikely to happen, since all h_t have already been verified in step 4 to correctly classify at least 50% of the instances. It can only happen when a new database $\mathcal{D}_{k+1}$ is introduced. If $E_t < ½$, then the composite normalized error is computed as

$$B_t = E_t / 1 - E_t \tag{5}$$

The weights $w_t(i)$ are then updated, which are used in computing the next distribution D_{t+1}, which in turn is used in selecting the next training and testing subsets, TR_{t+1} and TE_{t+1}, respectively. The distribution update rule constitutes the heart of the algorithm, as it allows Learn++ to learn incrementally:

$$w_{t+1}(i) = w_t(i) \times \begin{cases} B_t, & if \ \ H_t(x_i) = y_i \\ 1, & otherwise \end{cases} = w_t(i) \times B_t^{\,1-[\![H_t(x_i)\neq y_i]\!]} \tag{6}$$

According to this rule, if instance x_i $i=1,2,\ldots,m$, is correctly classified by the composite hypothesis H_t, its weight is multiplied by a factor of B_t, which, by definition, is less than 1. If x_i is misclassified, its distribution weight is kept unchanged. This rule reduces the likelihood of correctly classified instances being chosen into TR_{t+1}, and instead increases the likelihood of misclassified instances to be selected into TR_{t+1}. If we interpret instances that are repeatedly misclassified as difficult-to-learn examples (hard instances), and those that are correctly classified as easy-to-learn examples (simple instances), the algorithm focuses more and more on hard instances, and forces additional classifiers to be trained with them.

After T_k hypotheses are generated for each database $\mathcal{D}_k$, the final hypothesis (final classification rule) is obtained by the weighted majority voting of all hypotheses. The voting mechanism effectively fine-tunes the decision boundary from the rough estimates provided by each hypothesis.

The algorithm learns new information by generating additional classifiers, and at the same time, it retains the knowledge acquired earlier. Note that the weight update rule allows an efficient means of achieving incremental learning by concentrating predominantly on difficult to classify instances. This is particularly true when new classes are introduced by the new database, because all classifiers generated in the previous iterations will misclassify the instances from the new class. These instances will therefore

be misclassified by the composite hypothesis, and they will be forced into the next training dataset. A portion of correctly classified instances is also included in the next training dataset to ensure some balance between classes. Also note that the procedure would not work nearly as efficiently, if the weight update rule were based on the classification performance of h_t only (as AdaBoost updates its distribution) instead of the composite hypothesis H_t.

III. Results

The algorithm was tested on a variety of real world and synthetic datasets. Due to space limitations, only two sets of results are presented in this paper, and interested readers are referred to [4] for additional information.

A. Optical Handwritten Digit Recognition (OHDR)

This database consisted of 5620 instances of digitized hand written characters, from which 1200 instances were randomly selected for training. The database was obtained from the machine learning repository of University of California, Irvine, available at the web site [5]. The characters were numbers, 0 through 9, digitized on an 8x8 grid, creating 64 attributes. This dataset was used to evaluate Learn++ on incremental learning without introducing new classes. Training data were divided into six subsets, $S_1 \sim S_6$, each with 200 instances containing all 10 classes to be used in six training sessions. In each *training session* TS_k, only one of these datasets was used. For each training session k=1,2,…,6, thirty hypotheses were generated by Learn++, using a single-hidden-layer MLP with 30 hidden layer nodes and 10 outputs with an error goal of 0.1. Each hypothesis h_t (t=1,2,…,30) of the k^{th} training session was generated using a training subset TR_t and a testing subset TE_t, each with 100 instances drawn from S_k. Remaining instances constituted the validation dataset, $TEST_t$, and they were used to evaluate algorithm performance. Table 1 presents the results.

Table 1. Performance of Learn++ on OHDR database

Set	h_t (%)	TS_1 (%)	TS_2 (%)	TS_3 (%)	TS_4 (%)	TS_5 (%)	TS_6 (%)
S_1	55	94	94	94	93	93	93
S_2	53	---	94	94	94	94	93
S_3	51	---	---	95	94	94	94
S_4	53	---	---	---	94	94	94
S_5	56	---	---	---	---	95	95
S_6	58	---	---	---	---	---	95
TEST	**41.3**	**82**	**84.7**	**89.7**	**91.7**	**92.2**	**92.7**

The h_t column indicates the average performance of individual hypotheses on each training dataset. Weak learners provide little over 50% performances, which improve to well over 90% when the hypotheses are combined. Each column thereafter indicates the performance of Learn++ on the current S_k. The last row shows the classification performance on the validation dataset, which improved progressively as new data became available, indicating the incremental learning of new information.

B. Gas Sensing

This small, but extremely challenging database, was obtained from responses of six quartz crystal microbalances (QCMs) to various concentrations of five volatile organic compounds (VOCs), including ethanol (ET), xylene (XL), octane (OC), toluene (TL), and trichloroethelene (TCE). When QCMs are exposed to VOCs, the molecular mass deposited on their crystal surface alters their resonant frequency, which can be measured using a frequency counter or a network analyzer. By using an array of QCMs, each coated with a different polymer sensitive to specific VOCs, the collective response of the array can be used as a signature pattern of the VOC. However, QCMs have very limited selectivity, making the identification a challenging task.

The database consisted of 384 six-dimensional signals, half of which were used for training. This dataset was used to evaluate Learn++ when additional classes are introduced by new databases. The database was divided into three training databases $S_1 \sim S_3$ and one validation dataset, *TEST*. S_1 had instances from ET, OC and TL, S_2 had instances mainly from TCE (and a few from the previous three), and S_3 had instances from XL (and a few from the previous four). Only one dataset was used during each training session TS_k, k=1,2,3. Single-hidden-layer MLPs with 30 hidden layer nodes and error goals of 0.1 were used as weak learners. *TEST* set included instances from all classes. Table 2 presents the results.

Table 2. Performance of LEARN++ on gas sensing database

Set	TS_1 (%)	TS_2 (%)	TS_3 (%)
S_1	96.2	77.5	76.3
S_2	---	87.5	82.5
S_3	---	---	90.0
TEST	**60.78**	**70.1**	**88.2**

The classification performance on the validation dataset *TEST* improved as new data became available. Note that at the end of TS_1, the test performance was only 61%. This is expected, since the classifiers were only trained with three of the five classes, whereas the *TEST* dataset included instances from all five classes. The classification performance improved steadily, as new classes became available to the classifiers. Further details on VOC recognition using QCMs can be found in [4]. More information on this database and sample signals are provided at the web site [6].

These results demonstrate that LEARN++ successfully converts a classifier (such as a MLP) into an incremental learning algorithm, which otherwise suffers greatly from catastrophic forgetting.

IV. Discussion: Learn++ and Psycho-Physiological Models of Learning

Various models of learning have been developed during the past century, and Learn++ shares a surprisingly large number of notions with these models. In particular, many concepts in learning models have one-to-one correspondences in the Learn++ algorithm. These correspondences are described in the following paragraphs.

Learning is defined as the process by which knowledge is acquired, and it is intimately related to memory, the retention of acquired knowledge. For many years, neurophysiologists believed that learning and memory were exclusively handled by the cerebral cortex of the brain, however, it was later realized that different types of learning and memory are handled at different distinct regions of the brain [7]. Recall

that Learn++ generates different decision boundaries for different regions of the input space, which are learned by different classifiers. We can, therefore, interpret different classifiers generated by Learn++ as different areas of the brain where knowledge is stored. Furthermore, when a specific area of the brain is surgically removed, not all knowledge associated with that area is lost [8]. This indicates that even a specific knowledge is distributed, with some redundancy, to various regions of the brain. Learn++ also exhibits similar behavior: given adequate number of classifiers, removing one or more classifiers will not deteriorate the performance of Learn++ for any specific class. Note that, this is in contrast to conventional classifiers. For example, if one or few nodes of a trained MLP are removed, the network will fail to identify many instances.

According to a commonly accepted model, acquiring and storing knowledge has two distinct stages. Any new information acquired by the brain is processed and stored in "*short-term memory*", which has a very limited capacity. When similar information is repeatedly presented, this information is then processed and stored in "*long-term memory*", which has a larger capacity. However, the ability to recall information from the long-term memory also diminishes, though rather gradually in time, as new information is presented to the long-term memory [8]. Learn++ has obvious corresponding entities to short and long term memory models: individual classifiers serve as short-term memory, whereas the composite classifier obtained by the ensemble of individual classifiers constitute the long-term memory of the algorithm. As new information (e.g., new classes) is introduced to the algorithm, earlier classes will tend to be forgotten, unless instances from earlier classes are also periodically presented. In fact, the weight update rule of Learn++ tries to keep a delicate balance between new and old information (hard and simple instances), with the balance biased towards the new information, so that the new information can be efficiently learned while previously acquired knowledge is not lost. However, just like in the above mentioned model, if instances from a specific class are not shown to the algorithm for a large number of hypotheses, while instances from new class are presented, then some information related to earlier classes may be lost.

Learn++ is also closely related to psychological models of learning, such as *operant conditionin*g. According to this model, developed by psychologists Thorndike and Skinner, a behavior is learned and reinforced through reward and punishment [7, 8]. Learn++ employs a similar reward and punishment scheme: for each class and/or instance that is not sufficiently learned, Learn++ is forced to continue learning through retraining with the un-cooperating instances and classes, until they are learned. In the mean time, reinforcement is obtained for successfully learned instances and classes by occasionally reintroducing them.

Three other important concepts in operant conditioning learning are generalization, discrimination and extinction, all of which have corresponding entities in Learn++. Generalization refers to performing similar behavior in similar but different conditions, whereas discrimination is distinguishing similar but sufficiently different situations. Learn++ shares these concepts with most other classification algorithms, as all classifiers are designed to be able generalize sufficiently similar but different instances and discriminate similar but sufficiently different instances. Extinction is removal of a previously learned behavior in the prolonged absence of situations that originally gave rise to that behavior. Physiologically, this corresponds to loss of information saved in the long-term memory in the absence of repeated learning.

Finally, despite the conflicting views on whether new neurons are synthesized in time or we are born with all the neurons we will ever have, there is a consensus that long-term memory requires the synthesis of new proteins and the growth of new synaptic connections [8]. This probably constitutes the most striking association and the strongest link between Learn++ and the neurophysiological models. Every new training session involves generating a new classifier and combining this classifier with the previous classifiers, which can be interpreted as forming new synaptic links.

V. CONCLUSIONS

A new algorithm, Learn++, is introduced which allows any supervised classifier to learn incrementally from new data without losing previously acquired knowledge, in the absence of original data. Learn++ is also capable of learning new classes with new data. Initials results obtained using various databases demonstrate the feasibility of the algorithm. An interesting property of Learn++ is that it has one-to-one correspondences to many of the concepts in various models of learning. Future work includes evaluating Learn++ on various other classifiers, and using the weighted majority voting to estimate the confidence of the algorithm in its own decision, as well as investigating further associations between the algorithm and psycho-physiological models of learning.

REFERENCES

[1] R. Polikar, L. Udpa, S. Udpa, V. Honavar, "LEARN++: An incremental learning algorithm for multilayer perceptron neural networks," *Proceedings of ICASSP 2000*, vol. 6, pp. 3414-3417, 2000.

[2] R. Schapire, "The strength of weak learning", *Machine Learning,* vol. 5, pp. 197-227, 1990.

[3] Y. Freund, R. Schapire, "A decision theoretic generalization of online learning and application to boosting", *J. Comp. and Sys. Sci.*, vol.55, pp.119-139, 1997.

[4] R. Polikar, "Algorithms for enhancing pattern separability, feature selection and incremental learning with applications to gas sensing electronic nose systems" *Ph.D. Dissertatio*n, Iowa State University, August 2000. Also available at engineering.rowan.edu/~rpolikar/PhDdis.pdf

[5] http://www.ics.uci.edu/~mlearn/MLRepository.html.

[6] engineering.rowan.edu/~polikar/RESEARCH/vocdata.html

[7] L.R. Johnson, *Essential Medical Physiology*, Lippincott-Raven: Philadelphia, PA, 1998, ch. 61, pp. 801-812.

[8] E.R. Kandel, J.H. Schwartz, T.M. Jessell, *Essentials of Neural Science and Behavior*, Appleton and Lange: Norwalk, CT, 1997, ch.35-36, pp. 651-699.